REMARKS

I. INTRODUCTION

In response to the Office Action dated March 1, 2005, claims 1, 9 and 17 have been amended. Claims 1-24 remain in the application. Entry of these amendments, and re-consideration of the application, as amended, is requested.

II. CLAIM AMENDMENTS

Applicants' attorney amendments to the claims set forth above are intended to broaden the language of the claims, and were not required for patentability or to distinguish the claims over the prior art.

III. PRIOR ART REJECTIONS

A. The Office Action Rejections

In paragraph (4) of the Office Action, claims 1-24 were rejected under 35 U.S.C. §103(a) as being unpatentable over Sheard et al., U.S. Patent No. 6,208,345 (Sheard), in view of Green et al., U.S. Patent No. 6,854,107 (Green). Applicants respectfully traverse these rejections.

Applicants' attorney respectfully traverses these rejections, in view of the amended claims above and the arguments below.

B. The Applicants' Independent Claims

Independent claims 1, 9, and 17 are generally directed to developing multi-tier business applications. The computer-implemented system of claim 1 is representative, and comprises an Integrated Development Environment (IDE), executed by a computer, for creating and maintaining a multi-tier business application on a multiple tier computer network, wherein the IDE includes a Topological Multi-Tier Business Application Composer that is used by a developer to graphically create and maintain the multi-tier business application, the Composer includes a window and a palette, the palette contains graphical constructs representing tiers and components of the tiers that are used to create and maintain a graphical representation of the multi-tier business application in the window, and when creating the multi-tier business application, the developer decides on a number of tiers, identifies workstations and servers within each of the tiers, and defines processing performed by each tier and its components.

C. The Sheard Reference

Sheard discloses a visual data integration system architecture and methodology. The system architecture includes a transport framework that represents a technology-independent integration mechanism that facilitates the exchange of technology-dependent data between disparate applications. A visual interface facilitates the design, deployment, and runtime monitoring of an integrated information system implementation. An integrated information system is developed visually through use of the visual interface by dragging and dropping components within a canvas area of the interface. The components are graphical representations of various telecommunications hardware and software elements, such as information stores, processors, input/output devices and the like. Various components may be packaged together as business extension modules that provide specific business integration capabilities. Interconnections between components are graphically established using a mouse to define sources and destinations of specified data. An underlying configuration/runtime information framework operating above and in concert with the transport framework effectively transforms the graphical interconnections into logical or physical interconnections, which results in the contemporaneous generation of an integrated runtime system. Format neutral data meta-models are employed to model the input and output data requirements of disparate systems and system components so as to remove any cross-dependencies that exist between the systems and technologies implicated in a data integration project. The visual interface enables runtime control and analysis of the business information and system aspects of an integrated system implementation. Visual views onto the live deployment provide consistent management and control for system integrators, business integrators, system managers, and business managers using a single visual interface.

C. The Green Reference

Green describes a system and method for designing a software architecture for utilizing software components in building extensible N-tier software applications, the method comprising specifying a set of software component rules for creating software components; specifying a set of tier rules for creating tiers; and specifying a set of assembly rules further comprising association rules by which each tier may be associated with at least one software component and linkage rules by which each tier may be linked to at least one other tier. The tier rules may further comprise a set of association rules by which each tier created with the set of tier rules may be associated with at least one software component created using the software component rules; a set of tier framework rules

to provide an architected context for software components within a tier; and a set of package rules to provide for logical grouping of interfaces within a framework defined by the tier framework rules to provide a set of specific behaviors for the tier.

D. The Applicants' Invention is Patentable Over the References

The Applicants' invention, as recited in independent claims 1, 9, and 17 is patentable over the references, because it contains limitations not taught by the references.

The Office Action asserts that Sheard and Green together disclose these elements at the following locations: Sheard: Col. 3, lines 12-13, Col. 49, line 1 – col. 50, line 17, Col. 3, lines 16-18, Col. 3, lines 24-26, Col. 6, lines 11-13; and Green: Col. 1, lines 16-21, Col. 4, lines 56-62 and Col. 3, lines 14-16.

Applicants' attorney respectfully submits that the identified portions of Sheard and Green, taken in combination, do not render obvious Applicants' independent claims. For example, at the indicated locations, Sheard and Green merely disclose the following:

> Sheard: Col. 3, lines 12-29 (actually, col. 3, lines 12-44)
>
> The present invention is directed to a visual data integration system architecture and methodology. The system architecture includes a transport framework that represents a technology-independent integration mechanism which facilitates the exchange of technology-dependent data between disparate applications. A visual interface facilitates the design, deployment, and runtime monitoring of an integrated information system implementation.
>
> An integrated information system is developed visually through use of the visual interface by dragging and dropping component icons within a canvas area of the interface. The component icons are graphical representations of various data processing and telecommunications hardware and software elements. Various component icons may be packaged together in business extension modules to provide users with specific business integration capabilities.
>
> Interconnections between components placed in the canvas area are graphically established using a mouse so as to define sources and destinations of specified data. An underlying configuration and runtime information framework effectively transforms the graphical interconnections into logical or physical interconnections, which results in the contemporaneous deployment of an analogous integrated runtime system. Format neutral data meta-models are employed to model the input and output data requirements of disparate systems and system components so as to remove any cross-dependencies that exist between the systems and technologies implicated in a data integration project. The use of data meta-models in this manner effectively componentizes the systems of the data integration project, thereby permitting interconnections between system components to be established and modified using visual drag-and-drop and meta-model mapping metaphores.

Sheard: Col. 49, line 1 – col. 50, line 17

45. A computer readable medium tangibly embodying a program executable for visually implementing a data communications interface through which data passes, the data comprising informational content and a protocol, the medium comprising:

visually depicting a first data exchange component and a second data exchange component with which only the informational content of the data is transported therebetween;

visually linking the first data exchange component with the second data exchange component so as to visually define the data communications interface; and

transforming the visually defined data communications interface into a runtime deployment of the data communications interface using data definition models associated with the first and second data exchange components.

46. A computer readable medium tangibly embodying a program executable for visually implementing a data communications interface through which data passes, the data comprising informational content and a protocol, the medium comprising:

visually depicting a first data exchange component and a second data exchange component with which only the informational content of the data is transported therebetween;

visually linking the first data exchange component with the second data exchange component so as to visually define the data communications interface; and

verifying the utility of the visual link between the first and the second data exchange components using first and second data definition models, the first and second data definition models defining data input and data output requirements of the respective first and second data exchange components.

Sheard: Col. 6, lines 11-13 (actually, col. 6, lines 7-19)

In FIG. 1, there is illustrated a visual data integration architecture in accordance with an embodiment of the present invention. The system 30 shown in FIG. 1 provides a transport framework 33 and a visual interface 31 to facilitate the design, deployment, and runtime monitoring of an integrated information system comprising a number of disparate applications. In broad and general terms, the transport framework 33 provides a technology-independent integration mechanism that facilitates the exchange of technology-dependent data between disparate applications. The transport framework 33 enables reliable and scalable routing of information between dissimilar applications and technologies.

Sheard: Col. 23, lines 10-14 (actually lines 10-20)

In typical use, the user designs a data integration layout when the System Integration view is active by selecting various adapters and components displayed in the palette 530 of the visual interface 501. This is achieved by dragging selected adapters from the palette 530 and dropping them onto the canvas 540 using a mouse or other input device. This operation results in the creation of a new entry for the selected adapter in the project file and, additionally, results in the creation of an instance configuration file in the projects directory using a copy of the default configuration derived from the component configuration file.

Sheard: Col. 24, lines 55-67 (actually col. 24, line 51-col. 25, line 7)

The integration of data across multiple platforms and multiple workstations is coordinated through the use of a distribution planning facility. Activating the Xchange button 544 results in the presentation of a menu item which permits the user to invoke a distribution planning panel. The distribution planning panel 550, an embodiment of which is shown in FIG. 19, includes a panel 552 that provides a tree view of the network environment currently in operation for a selected data integration project. Each node in the first level 554 of the tree represents the name of a project. The second level nodes 556 under the project nodes 554 indicate the names of the workstations on which specified components are operating. A third level of nodes 558 indicates the various components operating on a particular workstation. A fourth level of nodes 560 indicates details of either component or queue elements defined on the third level of nodes 558. For example, the components shown in panel 552 of FIG. 19 includes six individual adapters, namely, CGI, ODBC, Mail, Printer, Pager, and Monitor adapters. The Monitor adapter represents a monitoring process node that is typically distinguished from other adapter nodes in terms of color or font. It is noted that the network file system mapping used to access remote machines is typically set by a system administrator outside of the visual interface environment.

Sheard: Col. 29, lines 32-36 and 56-60 (actually, col. 29, line 32 – col. 30, line 26)

As was discussed previously, a meta-model approach is used to provide a system wide specification of object and contained attribute definitions that can be used to illustrate object layout, instantiate objects, and provide for translation from one meta-defined class to another. Each adapter accepts data in a specific defined meta-model definition, manipulates the data, and produces output data in a new meta-model definition. By comparing the input and output meta-models of two interconnected adapters, it is possible to determine whether the data exchanged between the adapters is valid. Minor inconsistencies in the data requirements of two communicating adapters may be adjusted by defining mappings between the two data meta-models. Severe incompatibilities between meta-model definitions are indicative of more fundamental data issues that may require some degree of redesign to correct. The use of a meta-model approach allows the validity of a data integration implementation to be verified, errors to be highlighted, and problems to be corrected.

Storage of the meta-model is typically implemented using a file based approach which advantageously removes any dependency on a particular database technology. Each object definition is contained in a separate file in order to isolate its definition and eliminate confusion between multiple object definitions. Each meta defined class is stored in a separate file which is named using a class plus some extension convention. The contents should be displayed in as flat a structure as possible. Each attribute consists of a single line which includes its name, type, and behavioral characteristics. Each line representing an attribute may conform to the following layout:

NAME | DX_DATATYPE | REQUIREMENT | RANGE (optional) | DefaultValue (optional)

By way of further example, a sample configuration for an object class named Customer is provided below:

```
CustomerName|DX_STRING|MANDATORY|256|
Bank|DX_STRING|MANDATORY|256|"Rich's Bank"
AccountNumber|DX_INTEGER|MANDATORY|0-9999999|
Balance|DX_REAL|OPTIONAL||0
```

The following example is provided using the object class Customer defined above:

```
CustomerName|DX_STRING|MANDATORY||
AccountList|DX_LISTOBJECT|MANDATORY||
BEGIN:
    CheckingAcct|DX_COMMONOBJECT|OPTIONAL||
    BEGIN:
    AccountNumber|DX_INTEGER|MANDATORY|0-9999999|
    Balance|DX_REAL|OPTIONAL||0
    END:
    SavingsAcct|DX_COMMONOBJECT|OPTIONAL||
    BEGIN:
    AccountNumber|DX_INTEGER|MANDATORY|0-9999999|
    Balance|DX_REAL|OPTIONAL||0
    END:
    MoneyMktAcct|DX_COMMONOBJECT|OPTIONAL||.
    BEGIN:
    AccountNumber|DX_INTEGER|MANDATORY|0-9999999|
    Balance|DX_REAL|OPTIONAL||0
    END:
END:
```

Green: Col. 1, lines 16-21

1. Field of the Invention

The present invention relates to software design of software architectures and, in particular, to the design of a software component architecture for the development of extensible tier software component applications, including compiled, interpreted, and on-the-fly applications.

Green: Col. 3, lines 14-16

GUID Globally unique identifier, e.g. a number having a predetermined number of bits that uniquely identifies a software component

Green: Col. 4, lines 56-62

In one embodiment, the present invention provides rules to define and create a particular N-tier architecture with a specified, initial number and type of tiers 30 and with a specified interface architecture for each tier 30, where each initial tier 30 satisfies one of a major portion of system functionality, such as business logic (processing), data, and the like.

The above portions of Sheard and Green do not teach or suggest deciding on the number of tiers, identifying workstations and servers within each of the tiers, and defining processing performed by each tier and its components, in the context of an Integrated Development Environment (IDE), that includes a Topological Multi-Tier Business Application Composer, which is used by a developer to graphically create and maintain a multi-tier business application, and includes a window and a palette, the palette contains graphical constructs representing tiers and components of the tiers that are used to create and maintain a graphical representation of the multi-tier business application in the window.

Instead, the above portions of Sheard merely describe a visual data integration system for visually linking data exchange components so as to visually define a data communications interface, while the above portions of Green merely describe the design of a software component architecture for the development of extensible tier software component applications.

Thus, the combination of Sheard and Green does not render obvious Applicants' claimed invention. Moreover, the various elements of Applicants' claimed invention together provide operational advantages over the combination of Sheard and Green. In addition, Applicants' invention solves problems not recognized by the combination of Sheard and Green.

Applicants' attorney submits that independent claims 1, 9, and 17 are allowable over the references. Further, dependent claims 2-8, 10-16, and 18-24 are submitted to be allowable over the references in the same manner, because they are dependent on independent claims 1, 9, and 17, respectively, and thus contain all the limitations of the independent claims. In addition, dependent claims 2-8, 10-16, and 18-24 recite additional novel elements not shown by the references.

IV. CONCLUSION

In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited.

Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicants' undersigned attorney.

Respectfully submitted,

GATES & COOPER LLP
Attorneys for Applicant(s)

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: April 29, 2005

By: _____
Name: George H. Gates
Reg. No.: 33,500

GHG/